SOA4All, enabling the SOA revolution on a world wide scale

Conference or Workshop Item

SOA4All, Enabling the SOA Revolution on a World Wide Scale

John Domingue
Knowledge Media Institute, The Open University
j.b.domingue@open.ac.uk

Dieter Fensel
University of Innsbruck
dieter.fensel@sti2.at

Rafael González-Cabero
Atos Research & Innovation, Atos Origin
rafael.gonzalez@atosresearch.eu

Abstract

SOA4All will help to realize a world where billions of parties are exposing and consuming services via advanced Web technology. The outcome of the project will be a comprehensive framework and infrastructure that integrates four complimentary paradigm-shifting technical advances into a coherent and domain independent service delivery platform: Web principles and technology as the underlying infrastructure for the integration of services at a world wide scale; Web 2.0 as a means to structure human-machine cooperation in an efficient and cost effective manner; Semantic Web technology as a means to abstract from syntax to semantics as required for meaningful service discovery; and context management as a way to process in a machine understandable way user needs that facilitate the customization of existing services for the needs of users.

1. Introduction

One of the most visible trends that has been gaining momentum in IT technologies field in recent years is the concept of service-oriented architectures (SOA). Service-orientation is a design paradigm that seeks the separation of concerns by founding the building blocks of a solution on top of services. Thus SOA specifies a set of loosely coupled services – whose interfaces are published, discovered and invoked over the Internet - along with their interactions. SOA is currently enjoying massive adoption in large corporations since it promises to close the gap between what companies require and what IT is able to deliver. Moreover, SOA promises a more flexible IT infrastructure that is able to react to business changes more quickly than the classic monolithic IT systems. Finally SOA is built on top of Web standards, making interaction among companies easier; even facilitating the formation of business ecosystems able to traverse enterprise boundaries.

Nevertheless despite of its benefits, SOA is being primarily used for internal integration and far less for external consumption. In particular companies seem still reluctant to expose their business services thorough the Internet. At the moment very few companies offer their (internal) services to others, e.g. customers on their own. This situation is changing however, as increasingly IT companies such as Yahoo, eBay, Amazon and Google are exposing their IT services over the Web; allowing others to integrate these services in order to build new applications in so-called "mash-ups."

Parallel to the emergence of SOA as a valid infrastructure alternative for large enterprises, the Web has continued its success and has become the dominant information medium for consumers and for the entire range of companies. It has helped many small and medium enterprises to be globally visible in a world dominated by large enterprises. Consumers have been dazzled by new means of participation brought forward by Web 2.0 technologies. Technologies that further simplify user contributions such as blogs and tagging communities have unleashed the power of cooperation with efforts such as Wikipedia[1] demonstrating the

[1] www.wikipedia.org

"wisdom of crowds" in creating the world's largest encyclopedia.

Today, the Web contains just around 25,000 Web services - a minuscule amount in comparison to the 30 billion Web pages constituting its content. In consequence, SOA is largely still an enterprise specific solution exploited by and located within large corporations as part of their in-house supply chains. Nevertheless, complex mobile devices and more efficient wireless communications facilitate ubiquitous computing; and as optical and broadband communication infrastructures expand, we expect the number of Web services to grow exponentially in the next few years. In particular:

- More companies will publish their offerings as services accessible through the Web inspired by the success of early adopters companies (e.g. Amazon).
- Web 2.0 has popularized concepts such as mash-ups and syndication though, for example, technologies such as RSS, Atom. They have thereby illustrated comparatively simple means for business networking and business flexibility.
- Efforts to turn the Web into a general platform for accessing and interconnecting arbitrary devices and services are maturing.

Hence, there is a need to master the very large, we must be able to handle the complexity of these systems confidently as well as provide them with learning capabilities and self-organizing functions. Crucially, systems and software must be secure, robust, dependable and optimized in terms of functionalities to cater for multiple audiences.

In particular, we envisage that the combination of Semantic Web and SOA will lead to the creation of a "service Web"—a Web where billions of parties are exposing and consuming billions services seamlessly and transparently and where all types of stakeholders, from large enterprises to SMEs and singleton end users, engage as peers consuming and providing services within a network of equals. But, as was highlighted in [1], SOA will not scale without: properly incorporating principles that made the web scale to a world wide communication infrastructure; significant mechanization of service lifecycle activities (which includes location, negotiation, adaptation, composition, invocation and monitoring as well as service interaction requiring data, protocol and process mediation); and a balanced integration of services provided by humans and machines. In a service-oriented world, services must be discovered and selected based on requirements, then orchestrated and adapted or integrated. Solving these problems in a scalable and manageable manner is a major pre-requisite to realize a web interconnecting billions of services (as the current Web does for information sources).

In this paper we present the principles and rationale behind the SOA4All project, and outline how these principles will provide the means and methods for an internet-scale deployment and adoption of SOA infrastructures. We will begin by describing the SOA paradigm. Then we contrast SOA principles with the principles underlying the Web, the Semantic Web and Web 2.0. Later on we describe the challenges and requirements for integrating Web, Semantic Web and Web 2.0 principles with the SOA paradigm. Finally we briefly outline SOA4All, the project where all of these activities will be carried out, enabling the adoption of services-oriented computing up to Web scale.

2. SOA and Service-orientation principles

A Web service is an interface that describes a collection of operations that are network-accessible through standardized Web protocols, and whose features are described using a standard XML-based language [2]. As such, as interfaces, Web services have proven very popular in industry, becoming a layer of abstraction over legacy hardware and software platforms. Thus Web services have already proven to be a cornerstone technology for EAI scenarios.

Even though, we consider that service-orientation provides a broad design paradigm beyond integrating legacy systems, permitting the separation of concerns utilizing services as the basic building blocks of functionality. Service-oriented computing represents a new generation of distributed system that encompasses its own design paradigm and design principles, design pattern catalogs, pattern languages, a distinct architectural model, and a set of associated technologies and frameworks [3]. Service-orientation provides a way of thinking about computational infrastructures in terms of services, service-based development and the outcomes of those services.

As we have already stated, this architectural model aims to enhance efficiency, agility, flexibility and productivity by positioning services as the primary atomic functional elements. In the context of our work we classify these services according to:

- **The functionality they provide within the architecture**. We distinguish between business and middleware services. Business services are services which various service providers supply through their back-end systems. Additionally, they are the subject of integration and interoperation within the architecture and can provide a certain value for users (such as booking a hotel room). On the other hand, middleware services are the main

facilitators for the integration and interoperation of business services (providing discovery and interoperability support).

- **The abstraction level within the architecture**. Namely, we distinguish between Web services and services. A Web service is a general service that might take several forms when instantiated (such as purchasing a flight), whereas a service is an actual instance of the Web service that is consumed by and provides a concrete value for a user (such as the purchase of a particular flight from Innsbruck to Vienna).

The SOA design paradigm thus captures a distinct approach to the analysis, design, and implementation to all types of service-oriented IT environments, introducing a set of principles which govern aspects of communication, architecture, and processing logic. According to [3] these design principles are: the Standardized Service Contract Principle; Loose Coupling Principle; Abstraction Principle; Reusability Principle; Autonomy Principle; Statelessness Principle; Discoverability Principle and Composability Principle.

2.1. Standardized Service Contract Principle

In order to make the description of service capabilities understandable to any interested party, the properties of a service should be compliant with some design standard, the service contract. The service contract may include any information regarding the identification of the services (e.g. URL, name, textual description); functional properties, such as the type of the input/output parameters, interaction model; and non-functional properties, such as QoS, the location of the service, security constraints, etc.

Standardization supports the interpretability of services, resulting in an increase in the predictability of the service behaviour. The ability to predict the future behavior of a service is a key mechanism to achieve scalability, since it allows the evaluation of the necessary computational resources required to enact a specific service. This mechanism enables the intelligent provisioning of resources to prevent software resources running out.

2.2. Loose Coupling Principle

The Loose Coupling Principle states that the interface of a service should impose low consumer coupling and should also be orthogonal to its surrounding environment. Loose coupling, as presented in [4], intentionally sacrifices precision in the description of the interfaces of services for a greater good: the achievement of flexible interoperability among systems which are heterogeneous with respect to technology, location, performance, and availability. Loosely coupled applications aim to be more reusable and adaptable to new requirements.

Loosely coupled systems, such as event-driven systems [5] or space-based systems [6] have proven to be highly scalable when compared with tightly coupled systems.

2.3. Abstraction Principle

The Abstraction Principle dictates that the details of software artifacts which are not indispensable for others to effectively use it should be hidden. Therefore all the information necessary to invoke the service is contained in the service contract; and all the knowledge of the underlying logic, technology, etc. should be completely buried. This principle can be considered as a synonym of the old software engineering concept of black boxing.

The Abstraction Principle enables replaceability, which as outlined in [7], combined with fault isolation and fault recovery, enhances scalability.

2.4. Reusability Principle

The Reusability Principle states the functionality provided by services are as domain and context independent as feasible, facilitating reuse [3]. As a direct consequence of the application of this principle the logic of a service should be highly generic, independent from its original usage scenario. The Reusability Principle is a key enabler for SOA infrastructures, since it makes possible the creation of huge libraries of domain-independent services that leverage the construction of new complex context-dependent services.

2.5. Autonomy Principle

The Autonomy Principle states that services should be able to carry out their processes independently from outside influences. The only way to affect the results of a service should be through the modification of the input parameters as specified in the service contract.

Service autonomy increases reliability and more importantly predictability and fault isolation, which as presented in [7] leads to an increase of the overall system scalability.

2.6. Statelessness Principle

The Statelessness Principle dictates that services should minimize resource consumption by deferring the management of state information when necessary [3]. This notion of statelessness has been taken to the extreme by the REST architectural style [8], which has also been successfully applied to SOA in recent years.

Conformance with the Statelessness Principle is vital for the scalability of the entire SOA infrastructure, since state maintenance is one of the most resource consuming tasks in computer science. A small reduction of the amount of state information taken in to account by each service produces a significant reduction in the resources used by the overall system.

2.7. Discoverability Principle

The Discoverability Principle states that we should annotate services with metadata to enable their discovery by interested parties. This principle is closely related with the Standardized Service Contract Principle.

2.8. Composability Principle

The Composability Principle identifies services as effective composition participants, regardless of the size and complexity of the composition [3]. From a bottom-up perspective, we consider combining simpler services into larger services; from a top-down view, service composition is an effective way to tackle with the complexity of certain types of processes.

The Composability Principle is a core element within the definition of a service Web, since the ability to create new services easily is a key pre-requisite to the widespread take up of SOA.

3. Enhancing SOA with the Web principles

The Web is based also in a collection of principles that lead to a highly scalable means for electronic publication. We believe that we should analyze and apply these principles to service-orientation, which will lead to a global, dynamically changing environment of services accessible for third-party usage. Within this environment, services will undergo many changes; and there will be a very high churn rate. Users and resources will appear, disappear, and change location; resources can be initially free, and then transform to pay-per-use; and occasionally be blocked, out of service, or inspected for antitrust violations, etc..

The provision of Web-based lightweight integration infrastructures will facilitate openness and easy adoption for both the service provider and consumer. Moreover, the adoption of Web principles will open service-orientation beyond the boundaries of single organizations. We advocate several main Web principles for widespread acceptance.

We believe that the transformation of SOA into an architecture comprised of billion of services requires the embodiment of the principles which made the Web such a successful platform for the worldwide sharing of content. The major principles we will incorporate to SOA4All from the Web are described in the following subsections.

3.1. Distributed Principle

The Distributed Principle is the process of aggregating several computing entities' power to collaboratively run a single task, transparently and coherently, so that those entities appear as a single centralized system. Applying this principle to the architecture middleware system will allow the transparent distribution of components over the network so that executing processes running in a middleware can be scaled across numerous physical servers over a network. The distributed principle would also apply to business services, enabling running processes to span across enterprises distributed over a network.

3.2. Openness Principle

The Openness Principle states that a system should be easy to extend; in principle everybody should be able to contribute effortlessly to the system either as a provider or consumer of information. The usage of this infrastructure as a service provider or user must be as simple, smooth, unrestricted and even as possible.

Openness is a major and essential necessity to ensure global adoption.

3.3. Interoperability Principle

Interoperability should be provided through the integration of heterogeneous proprietary and legacy solutions through a common interface. Interoperability on the Web is platform and vendor neutral allowing all providers and requesters of information to participate on level playing field.

3.4. User-centric Principle

The User-centric Principle puts the user in the centre of the architecture. This principle is associated with concepts such as personalizing business services, facilitating service usability, promoting multichannel access and service delivery, building trust, and achieving efficiency, accountability, and responsiveness according to user requirements. It will also facilitate the seamless implementation of business processes across organizational boundaries.

4. Enhancing SOA with semantic technologies

Current standards for describing Web services use syntactic (XML-based) notations such as WSDL. As these descriptions are not machine readable, IT staff must carry out all of the tasks associated with creating and maintaining Web service-based applications. The requirement for specialist workers to be involved in all points in the Web service lifecycle causes numerous problems, the most significant of which is the lack of scalability. Maintaining millions of services, let alone billions, to cope with environmental and context changes solely through human effort is simply not feasible.

Tasks such as Web service discovery, composition, and invocation can be automated to a great extent by applying semantic technologies (such as OWL-S [9], WSMO [10], WSDL-S [11]). For example, semantics allows programs to access services through a machine-processable description of offered capability rather than as an endpoint. The use of semantics thus forms a scalable access layer over Web service data and processes.

4.1. Semantic Principle

In short the markup of a Web service with formal descriptions makes them computer-interpretable, use-apparent and agent-ready [12]. The combination of semantics with service-orientation allows us to define scalable, semantically rich, formal service models founded on ontologies.

A semantic approach to the modeling and implementation of service-based applications will facilitate the intelligent governance of SOA environments. Semantics will enable the management of categories of services as a whole; aiding the user in the visualization and update of services; facilitating the (semi) automation of service lifecycle activities, such as service discovery, contracting, negotiation, mediation, composition, and invocation; and enabling the advanced monitoring of execution and provenance analysis associated with the enactment of millions of services.

5. Integrating SOA with the Web 2.0 and Semantic Web

Embedding the principles underlying the Web, the Semantic Web and Web 2.0 within SOA will require the following in order to bring the SOA4All vision to reality:

- The Web principles and technology have to be applied to service-orientation creating an open and dynamically changing environment of services amenable to third-party usage.
- We have to provide Semantic Web technology as a means to implement a scalable access layer, both to data and processes
- We have to make usage of Web 2.0 technology as means to generate and access the semantic service layer.

The above objective of integrating all the principles generates the following requirements and challenges which will be addressed in SOA4All

5.1. Worldwide access mechanisms

If we wish realize a wide-spread SOA infrastructure we require a worldwide communication infrastructure able to deliver:

- **A global addressing schema,** which in simplest form may be a unique name and, more elaborately, a description of the capability of a service (that is, the degree to which it can be used to achieve a certain goal).
- **A transport layer** to transmit requests for and the results of service invocations.
- **A platform-independent interface** to process (client) service requests and access to service implementations. The trade-off though, as pointed out in [8], is that a uniform interface degrades efficiency, since information is transferred in a standardized form rather than one which is specific to an application's needs

5.2. Lightweight and new ways of semantic provisioning

Web service formal descriptions should be leveraged in order to allow lightweight reasoning about services; and they should also be extended to describe new forms of services. More precisely we should provide:

- **Lighter representations and representational languages.** Current initiatives (such as OWL-S

and WSMO incorporating languages such as WSML[13]) are powerful but are relatively complex to use; computationally lighter alternatives open the possibilities for wider take up, based, for example, on RDF(S) [14] and SAWSDL [15].

- **Extend to reflect the new mechanisms available in Web 2.0.** Within Web 2.0, there are many services in the form of mash-ups, gadgets and pipes which do not use standard Web services technology for their interface description, communication or enactment. These function provider entities should also be considered and accordingly annotated.

5.3. Decentralized changeability and dynamicity

Content can appear, be modified, or disappear in an *ad hoc* fashion. That is, the provisioning and modification of content should be under the distributed control of peers rather controlled by a central authority. Central control would hamper access and therefore scalability.

5.4. Automation of the activities that cover the entire services lifecycle

If we want to handle in an effective manner an environment composed by a huge number of decoupled services, we should provide highly automated governance mechanisms including:

- **Automatic service location**. This includes sub-tasks such as Service Crawling for collecting Web service information from the Web; Semantic Indexing to allow intelligent queries on top of the Semantic crawling solution; Service Discovery to allow users to find new services matching their needs which have to be formalized in some way by the user; Service Selection provides means to select the most suitable service among those services discovered by utilizing a service ranking algorithm; and Service Adaptation, in order to refine the results of discovery through interaction with the services and/or their providers, tailored according to users' needs.
- **Automatic Web service invocation.** The invocation of Web Service is usually hard-coded within client applications. In contrast, automatic Web service invocation is the automatic invocation of a Web service by a computer program or agent, given only a declarative description of the target service.
- **Automatic Web service construction**. This task involves the automatic selection, composition, and interoperation of Web services to perform some complex tasks, given a high-level objective.

5.5. Automation of central mechanisms to route requests or responses

Nowadays most of the routing activities are mainly carried out either by routing tables or rule engines based on solutions such as XPath and XQuery. These solutions are inherently non-scalable, very costly to maintain, and can easily become outdated. In order to create systems based on services that might appear and disappear continuously, we need more dynamic mechanisms. More importantly, all these routing techniques rely and base their decisions solely on syntax-based properties of requests and responses. We need to propose efficient semantic-based routing techniques that provide smarter routing, resulting in a highly-scalable smart semantic middleware.

5.6. Enabling of n:m asynchronous interactions

We should enable new models of interaction which enhance scalability, and facilitate the broadcast or multicast of requests and responses seamlessly. The Web is based on anonymous distribution through publication, which we believe has be true of any infrastructure aspiring to scale to the same magnitude.

5.7. Blurring the distinction between roles and nature of interactions participants

From a purely technological viewpoint, the mechanisms used in Web 2.0 are similar to the "standard" Web. However, Web 2.0 brings a number of Web-related concerns to the fore which when incorporated into SOA will aid in the creation of a Service Web:

- **Blurring the distinction between content consumer and content provider** – within Web 2.0 the provision of content has been democratized – in contrast to the standard Web where users are considered to be passive spectators of read-only content.
- **Blurring the distinction between service consumer and service provider.** The classic client-server model of interaction no longer reflects the nature of the Web and thus we should introduce richer models of interaction.
- **Blurring the distinction between machine and human-based computation.** Web 2.0

technologies provide a means to generate and access the semantic service layer outlined above. Incorporating human interaction and cooperation in a comprehensive fashion creates a route to solving tasks such as service ranking and mediation, which otherwise remain computationally infeasible. In several scenarios, Web 2.0 and human computing approaches, together with their underlying social consensus-building mechanisms, have proven the potential of combining human-based services with services provided via automated reasoning. In our view the transparent provisioning of services abstracting over whether the 'engine' is a human or machine will significantly increase the overall quality of services available to the end-user. In the end, a service need not necessarily by supplied by a computer program, enabling for example, current approaches to service discovery and (human) expert finders to be combined.

6. SOA4All, a Web of billions of Services

SOA4All will help to realize a world where a massive number of parties expose and consume services via advanced Web technology. The outcome of the project will be a comprehensive framework and software infrastructure which will integrate four complimentary advances into a coherent and domain independent worldwide service delivery platform. To achieve such an scalable and widely adopted infrastructure and framework, SOA4All stands on the four main pillars which are depicted in Figure 1:

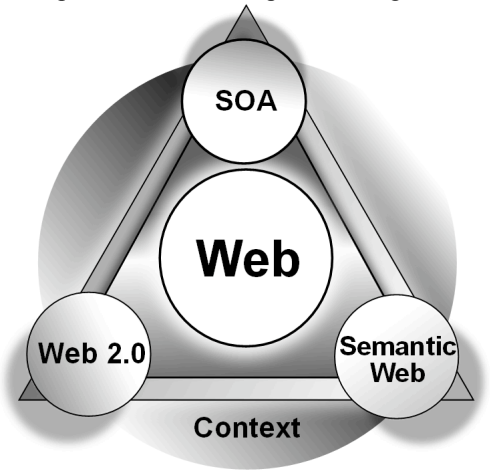

Figure 1 Cornerstones of SOA4All.

– **Web principles and technology as the underlying infrastructure** for the integration of services at a world wide scale. We will apply all the principles identified earlier to scale SOA to a world communication infrastructure. Moreover, we consider the Web as the underlying infrastructure to which we add machine-interpretable layers.
– **Web 2.0 as a means to structure human-machine cooperation** in an efficient and cost effective manner. SOA4All will make use of Web 2.0 technology as a means to generate and access the semantic service layer. Including human interaction and cooperation will enable us to provide solutions to certain classes of task such as service ranking or mediation that otherwise remain unfeasible. Web 2.0 and human computing approaches together with their underlying social consensus building mechanisms have proven the potential of integrating human based services and services solved by automated reasoning. Web 2.0 will be used for human interaction, and also will be used in SOA4All to support a range of activities including the ranking of services, service deployment, the acquisition of context and the semi-automation of service composition.
– **Semantic Web technology as a means to abstract from syntax to semantics**. The intensive use of semantics within the services environment described by SOA4All will reinforce our approach to dependability and fault tolerance. This aspect will be addressed in the indexing, crawling, discovery and adaptive dynamic composition. The expected result of this research will be the on-the-fly substitution of services - when required services will be composed dynamically at run time and service faults will be resolved by finding the most suitable alternative service to achieve the overall result.
– **Context management as a mechanism to process user requirements in a machine understandable way**, facilitating the customization of existing services for the needs of users. SOA4All will include context from a global perspective and will enable the specification of some parameters related to security as part of the service related formal descriptions, establishing restrictions and requirements for the composition and deployment of services in terms of the individual user, the community or the target organization. In turn this will make our solutions more flexible, secure and robust.

7. A promising future

From our point of view, we believe that the successful integration of Semantic Web and service-oriented technologies will form the main pillar of the

software architecture of the next generation of computing infrastructure. We envision a transformation of the Web from a Web of static data to a global computational resource that truly meets the needs of its billions of users, placing computing and programming within a services layer to facilitate computing's real goal: placing problem solving in the hands of end users through a truly balanced cooperative approach.

8. Acknowledgments

The work is funded by the European Commission under the project SOA4All (FP7- 215219).

9. References

[1] R. Benjamins, J. Davies, E. dorner, J. Domingue, D. Fensel, O. Lopez, R. Volz, A. Wahler, and M. Zaremba, "Service Web 3.0", STI Innsbruck Technical report. Available at http://www.sti-innsbruck.at/results/browse/technical-reports/details/?uid=35

[2] F. Curbera, W.A. Nagy, and S. Weerawana, "Web Service: Why and How?", In Proceedings of the OOPSLA-2001 Workshop on Object-Oriented Ser-vices. Tampa, Florida, 2001.

[3] T. Erl, *SOA Principles of Service Design*, The Prentice Hall Service-Oriented Computing Series from Thomas Erl, July 28, 2007

[4] D. Kayne, *Loosely Coupled, The Missing Pieces of Web Services* Rds Associates Inc, ISBN: 1881378241, 2003

[5] P.Th. Eugster, P. Felber, R. Guerraoui, Handurukande, "Event systems. How to have your cake and eat it too", 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02), 2002

[6] R. Krummenacher, E. Simperl and D. Fensel, "Towards Scalable Information Spaces", Workshop on New forms of reasoning for the Semantic Web: scaleable, tolerant and dynamic, ISWC 2007.

[7] J. Armstrong, *Making reliable distributed systems in the presence of software errors*. PhD thesis, Royal Institute of Technology, Swedish Institute of Computer Science (SICS), Stockholm, December 2003.

[8] R. T. Fielding, R. Thomas, *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, 2000

[9] D. Martin, M. Burstein, Hobbs J., O. Lassila, D. McDermott, S. McIlraith, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, "OWL-S 1.1 Release: Semantic Markup for Web Services", available at: http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2004

[10] D. Fensel, H. Lausen, A. Polleres, J. De Bruijn, M. Stollberg, D. Roman, J. Domingue. *Enabling Semantic Web Services: Web Service Modeling Ontology*. Springer, 2006.

[11] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.T. Schmidt, A. Sheth, K.Verma, WSDL-S Technical NoteVersion 1.0 Web Service Semantics, 2005

[12] S. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services", IEEE Intelligent Systems, 16(2):46–53, 2001.

[13] The Web Service Modelling Language WSML http://www.wsmo.org/TR/d16/d16.1/v0.21/

[14] RDF Vocabulary Description Language 1.0: RDF Schema W3C Recommendation 10 February 2004 http://www.w3.org/TR/rdf-schema/

[15] J. Farrell, H. Lausen, Semantic Annotations for WSDL and XML Schema W3C Recommendation 28 August 2007 http://www.w3.org/TR/sawsdl/